RECEIVED
CENTRAL FAX CENTER

DEC 14 2006

I HEREBY CERTIFY THAT THIS CORRESPONDENCE IS BEING FACSIMILE TRANSMITTED TO THE U.S. PATENT AND TRADEMARK OFFICE (FAX NO. (571) 273-8300) ON THE DATE INDICATED BELOW:

Date of Transmission: 12-14-06 By: Lisa M Hopkinson
Lisa Hopkinson

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of Daniel M. Lavery, et al. Atty. Docket No: 42390.P11237

App. Serial. No.: 09/886,585 Group Art Unit: 2191

Filed: 06/13/2001 Examiner: Rampuria, Satish

Title: METHOD AND APPARATUS FOR COMPILER-GENERATED TRIGGERING OF AUXILIARY CODES

Mail Stop: Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

BRIEF ON APPEAL

Pursuant to Appellant's Notice of Appeal filed on September 22, 2006, Appellant presents this Brief and fee under 37 C.F.R. § 1.17(c) in appeal of the Final Rejection dated June 22, 2006.

I. REAL PARTY IN INTEREST.

Intel Corporation is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES.

There are no related appeals or interferences before the Board of Patent Appeals and Interferences known to Appellant, the Appellant's legal representatives, or assignee that will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1-30 are pending in the application. Claims 1-30 stand finally rejected and are the claims subject to this appeal as are reproduced in Appendix A.

IV. STATUS OF AMENDMENTS

No amendments were filed after the Final Office Action dated June 22, 2006 (hereafter "Final Office Action").

V. SUMMARY OF CLAIMED SUBJECT MATTER

Simply stated and generally speaking, embodiments of Appellant's invention are directed to a method and apparatus for providing auxiliary computation (Specification, Page 2, lines 31-32). In auxiliary computation, an event may trigger the invocation and execution of an auxiliary code as a separate auxiliary thread. The auxiliary thread may execute concurrently with the original thread that triggered the invocation and execution of the auxiliary thread. (Specification, Page 2, line 32 – Page 3, line 2). Auxiliary threads may be spawned when encountering a "basic trigger", which may occur when a designated instruction in the non-auxiliary thread is processed, e.g., when the instruction

is retired, or by a "chaining trigger", when one auxiliary code explicitly spawns another. (Specification, Page 3, lines 4-7).

As embodied in independent Claims 1, 13, 18, 22 and 29, according to embodiments of the invention, initially, a "parent thread" executes normally. (Specification, Page 3, lines 16-17) At a specific time, a trigger may occur, e.g., when the parent thread receives an instruction that has been designated as a "trigger instruction". (Specification, Page 3, lines 17-18) Any type of instruction or subset of types of instructions may be treated as a trigger. Depending on the processor implementation, e.g., in a processor that uses associative lookup tables to interpret machine instructions, every instruction may be treated as a trigger instruction. (Specification, Page 3, lines 18-21) After the trigger instruction has been received, the parent thread then may execute instructions found in an auxiliary code associated with the trigger instruction. (Specification, Page 3, lines 22-23)

The instructions in the auxiliary code may be provided explicitly by the user or may be generated by the compiler or other application. (Specification, Page 3, lines 24-25) The auxiliary code may also be provided after the initial compilation, e.g., by a dynamic compiler that receives feedback regarding execution profiles of the original compiled function. (Specification, Page 3, lines 25-27) The instructions in the auxiliary code may be duplicates of selected instructions in the original function. These duplicated instructions need not be contiguous or successive instructions in the original function. (Specification, Page 3, lines 27-30)

The auxiliary code may be configured to include two parts, a stub and a body. The stub may include the instructions used to spawn an auxiliary thread. (Specification,

Page 3, lines 32- 33) The body may include the instructions, which are to be executed by the auxiliary thread. (Specification, Page 3, lines 34-35) Before spawning the auxiliary thread, the"parent" thread may first save its state information, e.g., by copying the values contained in the parent thread's registers to a predetermined scratch memory location. The parent thread may also test various conditions, e.g., hardware state information. (Specification, Page 3, line 35- Page 4, line 4)

A new, auxiliary thread may then be spawned by allocating a hardware thread context. If a free hardware thread context is not available, then the spawn request may be ignored, or alternatively the spawn request may be queued for later execution. (Specification, Page 4, lines 6-9) The auxiliary thread may receive all or part of the parent thread's state information. (Specification, Page 4, lines 9-10) The new, auxiliary thread may then begin execution of instructions provided in the body of the auxiliary code and while the auxiliary thread executes, the parent thread may continue to execute concurrently with the auxiliary thread. (Specification, Page 4, lines 16-18) Alternatively, the parent thread may stall and wait for the completion of the auxiliary code by the auxiliary thread. (Specification, Page 4, lines 22-23) Other execution schemes may also be provided, e.g., the parent thread might run in parallel until receiving a pre-specified signal, or wait until it receives a pre-specified signal from the auxiliary code and then resume execution in parallel. (Specification, Page 4, lines 23-26)

RECEIVED
CENTRAL FAX CENTER
DEC 14 2006

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The sole issue for consideration in this appeal is as follows:

A. Whether Claims 1-30 are properly rejected under 35 U.S.C. § 103 as being unpatentable over U.S. Patent No. 6,389,446 ("Torii") in view of U.S. Patent No. 6,754,888 ("Dryfoos").

VII. ARGUMENTS

A. THE COMBINATION OF TORII AND DRYFOOS IS IMPROPER

First and foremost, Appellants submit that the references cannot be combined in the manner suggested by the Examiner. Torii describes multi-processor system executing a plurality of threads simultaneously (Torii, Title), while Dryfoos describes a facility for evaluating a program for debugging upon detection of a debug trigger point (Dryfoos, Title). There is no suggestion or motivation in either reference for such a combination. As set out in M.P.E.P. § 706.02(j), "(t)here must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings." Appellants strongly disagree that that there is any such motivation in the present case.

The Examiner's alleged motivation is that it would have been obvious to combine Torii with Dryfoos to render other elements of the claimed invention unpatentable because "one of ordinary skill in the art would be motivated to select an entry in a trigger table to differentiate between instruction programs and instructions in the operating system's service routine as suggested by Dryfoos". Appellants respectfully submit that

this does not suggest a motivation, merely a result. There is no teaching in either Torii or Dryfoos to actually suggest this combination. The mere fact that the combination *may* provide an *advantage* does not *prima facie* mean that the combination is *obvious*. In the present case, there is no teaching in either reference to suggest that it would have been obvious to one of ordinary skill in the art to combine the references in the manner described by the Examiner.

The Examiner fails to address the Appellants' argument in the Final Office Action, stating simply that Appellants "only makes general allegations of improper hindsight reasoning and does not point out any errors in the rejection". As a result, the Examiner maintained that the rejection was proper. Appellants strongly disagree. Appellants respectfully submit that the Examiner bears the burden of establishing a *prima facie* case of unpatentability and failure to do so does not shift the burden to Appellants. As is well-established, in order to establish a *prima facie* case of unpatentability under 35 U.S.C. § 103, the combination of cited prior art must disclose every limitation of the claims being rejected. Therefore, if even one claim element or limitation is not taught by the reference, a *prima facie* case is not established. Additionally, as the Federal Circuit has noted,

"As adapted to *ex parte* procedure, Graham [v. John Deere Co.] is interpreted as continuing to place the 'burden of proof on the Patent Office which requires it to produce the factual basis for its rejection of an application under sections 102 and 103."

In re Piasecki, 745 F.2d 1468, 1472, 223 USPQ 785, 788 (Fed. Cir. 1984) (citing *In re Warner*, 379 F.2d 1011, 1016, 154 USPQ 173, 177 (CCPA 1967)). The Examiner thus has the burden of producing a factual basis for his rejection and for establishing unpatentability by identifying how each recited claim element is allegedly disclosed by the cited reference(s) or combination of references.

Appellants respectfully submit that the Examiner has failed to establish such a *prima facie* case (i.e., one based on factual basis) and has merely provided bare allegations that the claims are unpatentable over the combination of Torii and Dryfoos. Specifically, the Examiner fails to provide a factual basis for how or why those of ordinary skill in the art would combine these references. As previously stated, the Examiner's alleged motivation (i.e., the alleged factual basis for rejection) is merely a desirable result rather than a factual description of why one of ordinary skill in the art would combine these two references in the manner described. Appellants respectfully submit that simply selecting two random references which, if combined, may teach each element of the claims, does not rise to the level of establishing a *prima facie* case.

In summary, Appellants respectfully submit that the Examiner failed to provide a factual basis for the alleged motivation, thus failing to establish a *prima facie* case of unpatentability. The Examiner's statement essentially attempts to shift the burden on Appellants to establish the error in the rejection. While Appellants fully recognize Appellants' responsibility to refute the Examiner's *prima facie* case, Appellants respectfully submit that such a responsibility only arises after the Examiner has met his burden of establishing a *prima facie* case. Appellants thus respectfully contend that

shifting the burden onto Appellants at this time is premature because the Examiner has not yet established a *prima facie* rejection. Appellants thus respectfully maintain that the combination of these references is improper and requests the Board to overturn the Examiner's 35 U.S.C. § 103 rejections to Claims 1-30 for at least this reason.

B. CLAIMS 1-30 ARE PATENTABLE OVER TORII AND/OR DRYFOOS BECAUSE THE COMBINATION OF THESE REFERENCES FAILS TO TEACH OR SUGGEST CRITICAL ASPECTS OF THE CLAIMED INVENTION.

Independent Claims 1, 13, 15, 18, 22, 24 and 29 are method, system and article claims directed to embodiments of the invention. As such, these claims all include variations on the following elements: receiving an instruction, determining whether the instruction is a trigger instruction, selecting an entry in a trigger table if the instruction is a trigger instruction, the entry associated with the trigger instruction, and executing an auxiliary code referenced by the entry in the trigger table, the auxiliary code executing separate from the instruction. The Examiner collectively rejected Claims 1-21 based on the same rationale in the Final Office Action. Additionally, Claims 22-30 were rejected based on the same collective rationale. Appellant shall therefore address the rejections to these claims in two parts – Claims 1-21 and Claims 22-30.

Independent Claims 1, 13, 15 and 18 (Claims 1-21)

With respect to Claims 1-21, the Examiner concedes that Torii does not explicitly teach at least one significant element of the claimed invention, but suggests that this element is taught by Dryfoos. Appellants respectfully disagree. By way of example, the elements claimed in independent Claims 1, 13, 15 and 18 include the elements of receiving an instruction, determining whether the instruction is a trigger instruction, selecting an entry in a trigger table if the instruction is a trigger instruction, the entry associated with the trigger instruction, and executing an auxiliary code referenced by the entry in the trigger table. The Examiner essentially suggests that Torii teaches all but one element. While Appellants agree with the Examiner that Torii does not teach or suggest the element of "selecting an entry in a trigger table if the instruction is a trigger instruction, the entry associated with the trigger instruction", Appellants disagree that Torii teaches the element of "executing an auxiliary code referenced by the entry in the trigger table".

Appellants respectfully point out that the final element of Claims 1, 13, 15 and 18 ("executing an auxiliary code referenced by the entry in the trigger table") is in fact related to the element that the Examiner and Appellants agree is not taught or suggested by Torii. In other words, the element of "executing an auxiliary code referenced by the entry in the trigger table" is dependent on the previous element of "selecting an entry in a trigger table if the instruction is a trigger instruction, the entry associated with the trigger instruction". There is absolutely no mention in Torii of an auxiliary code referenced by the entry in the trigger table. The section of Torii highlighted by the Examiner states:

"For example, a thread generation instruction 2 generates thread #1 (1b) from thread #0 (1a)." (Torii, Col. 4, lines 32-33). Appellants fail to see how this sentence in Torii can be interpreted as "executing an auxiliary code referenced by *the entry in the trigger table*" (emphasis added).

In response to this argument, the Examiner in the Final Office Action once again attempts to make the same argument. Specifically, the Examiner states that "for the limitation 'executing an auxiliary code', the cited portion of Torii clearly reads on see col. 4, lines 32-33" (Final Office Action, Page 4). Torii, Col. 4, lines 32-33, as described above, reads as follows: "For example, a thread generation instruction 2 generates thread #1 (1b) from thread #0 (1a)." Appellants fail to see how this sentence remotely reads on the claimed element. The Examiner's rationale is that "the auxiliary code nothing but executing in parent child environment. The cited portion by the Examiner is executing the code in parent child environment." (Final Office Action, Page 4) Appellants strongly disagree. Appellant is not attempting to claim the general concept of threads running in "parent-child" environments. Read in context, the claim element is "executing an auxiliary code referenced by the entry in the trigger table, the auxiliary code executing separate from the instruction". This element is related to the previous element of "selecting an entry in a trigger table if the instruction is a trigger instruction, the entry associated with the trigger instruction". The Examiner's attempt to read this claim element without regard to the previous claim element is thus simply wrong. The crux of the invention lies in selecting an entry in a trigger table if the instruction is a trigger instruction, the entry associated with the trigger instruction and using the auxiliary code

referenced by the entry in the trigger table, where the auxiliary code executes separately from the instruction. The section of Torii highlighted by the Examiner as allegedly teaching the one element simply does not do so because this element appears to rely on an element taught only by Dryfoos. The Examiner's general allegation that this sentence reads on the claim element is thus facially deficient.

In summary since the Examiner concedes that Torii does not in fact teach the previous element of "selecting an entry in a trigger table if the instruction is a trigger instruction, the entry associated with the trigger instruction", the fact that Dryfoos may or may not teach this element (and Appellants maintain that it does not) is irrelevant because these elements cannot be read in the abstract, as the Examiner is suggesting. One element is dependent on the other. In other words, since the step of executing is dependent on the step of selecting, it would be impossible to combine the teachings of Torii and Dryfoos because Torii is not executing the auxiliary code referenced by the entry in the trigger table (i.e., the trigger table allegedly taught by Dryfoos). Appellants thus respectfully request the Board to overturn the Examiner's 35 U.S.C. § 103 rejection to Claims 1-21.

Independent Claims 22, 24 and 29

With respect to independent Claims 22, 24 and 29, the Examiner concedes that Torii does not explicitly disclose the element of "creating an entry in a trigger table, the entry associated with the trigger instruction and the auxiliary code". The Examiner

suggests, however, that Dryfoos teaches this element and that it would have been obvious to one of ordinary skill in the art to combine the two references. Appellants respectfully disagree.

Specifically, the section of Dryfoos highlighted by the Examiner makes no mention of creating an entry in a trigger table, the entry associated with the trigger instruction and the auxiliary code, as the Examiner suggests. First and foremost, there is no mention of a trigger table, merely a table. The table in Dryfoos contains "at least one of information identifying at lest some areas of main storage of the computing environment where programs to be debugged may reside, or information identifying at least one program of the computing environment to be excluded from debugging" (Dryfoos, Col. 1, lines 58-63). This is not, however, a "trigger table" as claimed. Additionally, Dryfoos merely mentions that the table contains "information", without any mention of creating entries for the table. As such, Appellants maintain that Dryfoos does not teach or suggest the claimed element and that independent Claims 22, 24 and 29 (and all claims dependent on these independent claims) are patentable over the combination of Torri and Dryfoos. Appellants thus respectfully request the Board to overturn the Examiner's 35 U.S.C. § 103 rejection to Claims 22-30.

VIII. CONCLUSION

It is respectfully submitted that in view of the foregoing, all of the pending claims (Claims 1-30) are patentable over the cited prior art references, alone or in any combination, and the Board is respectfully requested to overturn the rejections of record and allow this application to issue.

Respectfully submitted,

/Sharmini N. Green/

Sharmini N. Green
Registration No. 41,410
Intel Corporation
(714) 730-8225

c/o
Blakely, Sokoloff, Taylor & Zafman, LLP
12400 Wilshire Blvd., Seventh Floor
Los Angeles, CA. 90025-1026
(310) 207-3800

Date: December 14, 2006

APPENDIX A

1. (Previously presented) A computer-implemented method for executing a code, comprising:
receiving an instruction;
determining whether the instruction is a trigger instruction;
selecting an entry in a trigger table if the instruction is a trigger instruction, the entry associated with the trigger instruction; and
executing an auxiliary code referenced by the entry in the trigger table, the auxiliary code executing separate from the instruction.

2. (Original) The method of claim 1, further comprising:
spawning a new thread, the new thread executing instructions included in the auxiliary code.

3. (Original) The method of claim 2, further comprising:
executing the new thread concurrently with a parent thread, the parent thread including the trigger instruction.

4. (Original) A method for executing a code, comprising:
receiving a trigger instruction;
selecting an entry in a trigger table, the entry associated with the trigger instruction; and
executing a p-slice code referenced by the entry in the trigger table.

5. (Original) The method of claim 4, further comprising:
spawning a new thread, the new thread executing instructions included in the p-slice code.

6. (Original) The method of claim 5, further comprising:

executing the new thread concurrently with a parent thread, the parent thread including the trigger instruction.

7. (Original) The method of claim 6, further comprising:
storing state information from the parent thread before spawning the new thread.

8. (Original) The method of claim 7, further comprising:
copying the state information for use in the new thread.

9. (Original) The method of claim 6, further comprising:
storing a register value of the parent thread before spawning the new thread.

10. (Original) The method of claim 9, further comprising:
copying the register value of the parent thread for use in the new thread.

11. (Original) The method of claim 4, wherein the entry in the trigger table is selected by associative lookup of the trigger instruction.

12. (Original) The method of claim 4, further comprising:
reading an instruction pointer for the p-slice code from the entry in the trigger table.

13. (Previously presented) An article of manufacture comprising a computer-readable medium having stored thereon instructions adapted to be executed by a processor, the instructions which, when executed, define a series of steps to be used to control a method for executing a code, said steps comprising:
receiving an instruction;
determining whether the instruction is a trigger instruction;
selecting an entry in a trigger table if the instruction is a trigger instruction, the entry associated with the trigger instruction; and

executing an auxiliary code referenced by the entry in the trigger table, the auxiliary code executing separate from the instruction.

14. (Previously presented) The article of manufacture of claim 13, wherein the series of steps further comprises:
spawning a new thread, the new thread executing instructions included in the auxiliary code.

15. (Original) A system, comprising:
a current thread;
a function body configured to be executed as part of the current thread, the function body comprising at least one trigger instruction;
an auxiliary code; and
a trigger table, the trigger table comprising an entry, the entry associated with the trigger instruction and including a reference to the auxiliary code, the trigger table configured to allow the lookup of the entry when the trigger instruction is processed.

16. (Original) The system of claim 15, wherein the auxiliary code is configured to spawn a new thread when auxiliary code is executed.

17. (Original) The system of 16, wherein the auxiliary code is configured to store the value of a register associated with the current thread, when the auxiliary code is executed.

18. (Original) A system, comprising:
a current thread;
a function body configured to be executed as part of the current thread, the function body comprising at least one trigger instruction;
a p-slice code; and

a trigger table, the trigger table comprising an entry, the entry associated with the trigger instruction and including a reference to the p-slice code, the trigger table configured to allow the lookup of the entry when the trigger instruction is processed.

19. (Original) The system of claim 18, wherein
the p-slice code is configured to spawn a new thread when the p-slice code is executed.

20. (Original) The system of claim 18, wherein
the p-slice code is configured to store the value of at least one register associated with the current thread, when the p-slice code is executed.

21. (Original) The system of claim 18, wherein
the trigger table is an associative lookup table.

22. (Previously presented) A computer-implemented method for compiling, comprising:
receiving a function body, the function body comprising a trigger instruction;
outputting an auxiliary code associated with the function body and the trigger instruction; and
creating an entry in a trigger table, the entry associated with the trigger instruction and the auxiliary code.

23. (Original) The method for compiling of claim 22, further comprising:
creating a stub block, the stub block comprising a spawn instruction, the spawn instruction configured to spawn a new thread, the new thread configured to execute the auxiliary code.

24. (Original) A method for compiling, comprising:
receiving a function body, the function body comprising a trigger instruction;

outputting a p-slice code associated with the function body and the trigger instruction; and

creating an entry in a trigger table, the entry associated with the trigger instruction and the p-slice code.

25. (Original) The method of claim 24, further comprising:
receiving the p-slice code associated with the function body and the trigger instruction.

26. (Original) The method of claim 24, further comprising:
generating the p-slice code associated with the function body and the trigger instruction.

27. (Original) The method of claim 24, further comprising:
creating a stub block, the stub block comprising a spawn instruction, the spawn instruction configured to spawn a new thread, the new thread configured to execute the p-slice code.

28. (Previously presented) The method of claim 27, further comprising:
adding store instructions to the stub block, the store instructions configured to store state information of a current thread, the state information of the current thread including values contained in live-in registers of the new thread.

29. (Original) An article of manufacture comprising a computer-readable medium having stored thereon instructions adapted to be executed by a processor, the instructions which, when executed, define a series of steps to be used to control a method for compiling, said steps comprising:
receiving a function body, the function body comprising a trigger instruction;
outputting an auxiliary code associated with the function body and the trigger instruction; and
creating an entry in a trigger table, the entry associated with the trigger instruction and the auxiliary code.

30. (Original) The article of manufacture of claim 29, wherein the auxiliary code is a p-slice code.

EVIDENCE APPENDIX

None

RELATED PROCEEDINGS APPENDIX

None